Introduction to Computer Organization – Fall 2023

Homework 1

Due: @11:55pm, Monday September 25th on Gradescope

Name: Yuzhen Chen Uniqname: yuzhench

1. Submit a pdf of your typed or handwritten homework on Gradescope.

2. Your answers should be neat, clearly marked, and concise. Typed work is recommended, but not required unless otherwise stated. Show all your work, and state any special or non-obvious assumptions you make.

3. You may discuss your solution methods with other students, but the solutions you submit must be your own.

4. **Late Homework Policy:** Submissions turned in by 1:00 am the next day will be accepted but with a 5% penalty. Assignments turned in between 1:00 am and 11:55 pm will get a 30% penalty, and any submissions made after this time will not be accepted.

5. When submitting your answers to Gradescope you need to indicate what page(s) each problem is on to receive credit. The grader may choose not to grade the homework if answer locations are not indicated.

6. **The last two questions are group questions**.
   - **You will turn those questions in separately** and may do it in a group of up to two students (yes, you can do it by yourself if you wish).
   - It is an honor code violation if a student is listed as contributing who did not actually participate in working on that problem.
   - Further, we suggest that you not split this up but rather work on the problem as a group.

1. Multiple choice and the like. [15 points]

a. In the blank provided write the letter which best defines the associated term **[4]**

Compiler	A	A) ✓ converts high-level language to assembly or machine code
Loader	E	B) converts assembly ✗ language to high-level language
Assembler	C	C) ✓ converts assembly language to machine code
Linker	F	D) executes the instructions
		E) ✓ places machine code from hard disk to main memory
		F) ✓ merges object files into an executable

11

b. If each activation record for "combination" (stack frame for the function) requires 256 bytes of storage, and the maximum stack size possible on a machine is 3000 bytes, identify *all* of the following recursive function calls that will not be able to execute on the machine. **[6]**

```
int combination(int n,int r)
{
      if ((r==0) || (n==r))
           return(1);
      else
           return(combination(n-1,r)+combination(n-1,r-1))
           ;
}
```

A. combination(16,0) ✗
[✓ B.] combination(16,4) ✓
C. combination(8,4) ✓
D. combination(16,16) ✗
E. combination(12,6) ✓

BE <----- Answer(s) go here

c. Executing which of the following lines of LC2K assembly clears register 1 (that is sets register 1 equal to 0?) assuming register 0 is a zero? **[5]**

```
A.    nor    1 1 1  1' ✗
B.    nor    0 0 1 ✗
C.    lw     0 1 0 ✓
D.    .fill  1 ✗
E.    add    1 0 0 ✗
```

C <----- Answer(s) go here

0 nor 0 1

64 32 16 8 4 2 1

Problem 2 (15 points): Numbers and stuff

1. Convert the following numbers to 8-bit binary two's complement representation. **[4]**

   a. `34` (decimal) 0 1 0 0 0 1 0 / -64 32 16 8 4 2 1 0b 00100010

   b. `-100` (decimal) 1 0 0 1 1 1 0 0 / -128 64 32 16 8 4 2 1 0b 10011100

   -112 -104 -100

2. Convert the following 16-bit binary 2's complement numbers to decimal. **[4]**

   $9\times16^0+2\times16^1+14\times16^2+-4\times16^3 = 9+32+3584-16384 = -12759$

   a. `1100 1110 0010 1001` (-8 4 2 1 / -4, 14, 2, 9; 8 4 2 1) -12759

   b. `0000 1101 0001 0110` (8 4 2 1 / 0, 13, 1, 6; 8 4 2 1) 3350

   $6\times16^0+1\times16^1+13\times16^2 = 6+16+3328 = 3350$

3. Convert the following 16-bit binary numbers to hex. **[2]**

   A B C D E F / 10 11 12 13 14 15

   a. `1101 1000 1110 1011` (13, 8, 14, 11; 8 4 2 1) 0x D8EB

   b. `0010 1010 0011 1010` (2, 10, 3, 10) 0X2A3A

4. Provide the 32-bit representation of -12.75 as an IEEE floating point number. Provide your answer in hexadecimal. **[5]**

   12 0.75

   2|12 ...0
   2|6 ...0
   2|3 1
   1
   1100

   0.75×2
   1.5×2
   1.0
   11

   0.5+0.25.

   0XC14C0000

Problem 3 (15 Points): Moving data between memory and registers

For the following LEGv8 assembly code, indicate the final state of the registers and memory locations listed once this code has finished. Assume any memory address outside of that displayed in the initial memory state contains 0.

```
MOVZ    X0, #0x1001
LDURH   X2, [X4, #-2]
STURB   X3, [X0, #3]
LDURSW  X4, [X0, #-1]
STURH   X4, [X0, #1]
```

X4
X0 value+1
2 bytes
half
1bytes
half
Signed world. 4

X0 = 0x1001
X2 = X4 value

X4 = X0 value -1
= 0X1001 -1
= 0X1000.

A B C D E F
10 11 12 13 14 15

4177
+ 3
4213
000100...

X0 value+3 ← X3 value.
0X1001+3 ← 0X4177.
0X1004 ← 0X4177.

X0 value -1 ← X4 value.
0X1000 ← 0X1002

0X1001+1 = 0X1002.

Initial Register State		Initial Memory State	
Register	Value	Address	Value
X0	~~0x5~~ 0x1001	0x00001000	0x4
X1	0x1000	0x00001001	0x12
X2	~~0x6E~~ 0x1204	0x00001002	~~0x33~~ 0x04
X3	0x41FF	0x00001003	~~0x2E~~ 0x12
X4	~~0x1002~~ 0x2E331204	0x00001004	~~0xA1~~ 0xFF
X5	0x0	0x00001005	0x77

Fill in the following table. Express your final answers in hexadecimal. Assume memory is organized in a **little endian** manner.

Final Register State		Final Memory State	
Register	Value	Address	Value
X0	0x1001	0x00001000	0X4
X1	0x1000	0x00001001	0X12
X2	0x1204	0x00001002	0x04
X3	0X41FF	0x00001003	0X12
X4	0X2E331204	0x00001004	0XFF
X5	0x0	0x00001005	0X77

Problem 4 (15 points): Arrays

For problem 4 follow the C/C++ order of operations. All variables and arrays are initialized. You may not modify other registers.

Variable to register mappings *Starting addresses*

```
int a - r1        int b - r2        Go - r3        Blue - r4
```

Convert the following C code to 4 lines of LC2K assembly. Use r6 to hold the value of Blue[8], r7 to hold the value of the sum of b and Blue[8], and ~~r8~~ r6 (again) to hold the any results needed to compute Go[a+4]

$r_6 = r_4 + 8$

$\underbrace{\text{Go}[\overset{r_1}{a}+4]}_{r_3} = \underbrace{\overset{r_2}{b} + \underbrace{\text{Blue[8]}}_{r_6}}_{r_7};$

```
lw    4  6  8    // Blue[8] storein 6.
add   2  6  7    // r7 = r2+r6 = b+Blue[8]
add   1  3  6    // r6 = r3+r1 -> r6 = Go[a]
sw    6  7  4    // store b+Blue[8] to Go[a+4]
```

Problem 5 (15 points): Floating point--mind the gap

With integers, the "gap" between representations is always 1. This means that rounding a decimal to an integer can introduce an error of up to ∓0.5.

a) In a 32-bit IEEE floating point number, what is the smallest number greater than 2 that can be exactly represented? What is the gap between that number and 2? Briefly justify your answers. **[4]**

① Smallest number >2 that can be exactly represented:

$2 : 1.\underline{0} \times 2^1$ $\underbrace{0}_{\text{negative}}$ $\underbrace{10000000}_{\text{exponent}}$ $\underbrace{00\cdots\cdots000}_{\substack{\text{mantissa}\\23}}$ if we want to let it become a little bit bigger

then $\underbrace{\overset{1}{0}}_{\substack{\text{negative}\\1}}$ $\underbrace{\overset{8}{10000000}}_{\substack{\text{exponent}\\8}}$ $\underbrace{\overset{23}{00\cdots\cdots001}}_{\substack{\text{mantissa}\\23}}$. The gap is $2^{-23} \times 2 = 1 \times 2^{-22}$

b) Assuming we are always rounding to the nearest value we can exactly represent (rounding down in the case of ties), what is the largest value we can add to 2 and get a result that is still 2? **[3]**

max value is $\frac{1}{2} \times 2^{-24} \times 2^1 = 1 \times 2^{-24}$ since we can assume the 33th bit could be 2^{-24} and also we have 1 in exponent and we always rounding to the nearest value.

$$\overbrace{\underbrace{0}_{\substack{\text{negative}\\1}}\ 100}^{4}\ \overbrace{0001}^{1}\ \overbrace{0\ 100}^{4}\ \overbrace{110}^{C}\cdots\cdots 0$$

$\underbrace{10000010}_{\substack{\text{exponent}\\8}}$ $\underbrace{100110\cdots\cdots 0}_{\text{mantiss}}$

$(12.75)_{10} = (414C)_{hex}$

$1 \times 2^{-23} \times 2^{3} = 1 \times 2^{-20}$

c) Redo part a) but for 12.75 rather than 2. **[3]**

$12.75 \rightarrow 12,\ 0.75$

$2\,|\,12 \cdots 0$
$2\,|\,6 \cdots 0$
$2\,|\,3 \cdots 1$
1
1100

0.75×2
$0.5 \times 2 \cdots 1$
$0 \quad 0 \cdots 1$
11

$(12.75)_{10} = (1100.11)_2 \qquad 1.10011 \times 2^{3}$

d) Briefly explain how the gap between floating point numbers might cause a programmer difficulties. In particular, consider a for loop where small values are being added to a larger number many thousands of times. **[3]**

If we write a for loop and add a really small value to a very large number "a". Then, may the value of a will not change for several loop. since the gas is bigger than ~~the~~ update value and rounding the value down.

Problem 6 (15 points, group): Transistor Performance

You must **type** the answer to these questions. Read about Moore's Law and Dennard Scaling on Wikipedia (or some other source if you wish) then answer the following questions. Each answer should be under 100 words.

1. Define Moore's Law in your own words.

2. Define Dennard Scaling in your own words.

3. Imagine you have a microprocessor that follows Dennard Scaling perfectly. If you double the number of transistors on the chip while keeping the area constant, *roughly* what would you expect would happen to the performance and power consumption of the microprocessor? (We are looking for things like "go up", "go down" "stay the same" etc.) Briefly explain your reasoning.

4. Say that the power consumed by a single transistor is constant and you double the number of transistors on the chip while keeping the area the same. *Roughly* what would you expect would happen to the performance and power consumption of the microprocessor? (We are looking for things like "go up", "go down" "stay the same" etc.) Briefly explain your reasoning.

Problem 7 (10 points, group): Writing LC2K code

Consider the following C structure.

```
struct LinkedList {
          int element;
          struct LinkedList *next;
        };
```

The following LC2K2 code is supposed to sum the value of all of the elements in the linked list. The pointer of the last element of the linked list points to location zero, and no other pointer points to that location. Fill in the missing gaps with the correct values to cause this program to work. **[10 points]**

reg3 = (reg0 + start) = value.

pointe : r3

```
      lw   0    3    start  Load pointer to 1st element.

      add  0    0    4    reg4 = 0+0=0.  [sum=0 at beginning]

addLL beq  _3_   0    done

      lw   _3_   2    0      -> reg2 = current element value.

      add  (2)   4    4    reg4 = reg4+reg2  ----> reg4 stores [sum of element]

      lw   _3_   3    ___1___

      beq  0     0    addLL

done  halt

start    .fill  __first__

first    .fill  4          First element of the link list

         .fill  second     pointer to the next element

second   .fill  10

............                   (More of the data structure follows)
```